

InterPol: A Policy Framework for Managing Trust and Privacy in Referral Networks*

Yathiraj B. Udupi[†]
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
ybudupi@ncsu.edu

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
singh@ncsu.edu

ABSTRACT

Referral networks are a kind of P2P system consisting of autonomous agents who seek, provide services, or refer other service providers. Key applications include service discovery and selection, and knowledge sharing. This use of referrals is inspired by human interactions, where referrals are a key basis for judging the trustworthiness of a given service.

The use of referrals enable an agent to control how its request is processed, it also provides an architectural basis for four kinds of interaction policies. *InterPol* is a language and framework supporting such policies. InterPol has been implemented using a Datalog-based policy engine for each agent. It has been applied on scenarios from a (multinational) health care project. The contribution of this paper is in a general referrals-based framework for privacy and trust management, which is shown to effectively capture a variety of privacy and trust requirements of autonomous users.

Categories and Subject Descriptors

I.2.11[Artificial Intelligence]: Distributed Artificial Intelligence – *multi-agent systems*.

General Terms

Design, Languages.

Keywords

policy, trust, privacy, referrals, p2p.

1. INTRODUCTION

In an open distributed system, (discovering and) selecting among service providers is a key challenge. *Referral systems* are a less well-known but powerful kind of P2P system [1, 8]. Briefly, referral systems are multiagent systems

[†]The first author is a full-time doctoral student.

*We thank NSF (grant ITR-0081742).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AMAS'07, May 14–18, 2007, Honolulu, Hawai'i, USA.

Copyright 2007 IFAAMAS.

whose member agents follow a (generally, but not necessarily) cooperative protocol by issuing referrals to one another, thus sharing their knowledge about service providers and enabling improved service selection. However, traditional referral networks are difficult to engineer since they lack a declarative characterization of how the agents interact. The present paper offers a policy framework that would support easy administration based on a flexible and yet practical approach for agents to decide with whom to interact and how.

This paper describes *InterPol*, an implemented framework and a specification language for interaction policies in multiagent service networks. Policies capture requirements perspicuously and are used in many practical settings, such as for business or security. InterPol provides an application-independent vocabulary geared toward interaction policies in service networks. InterPol's novel features include capturing social primitives to capture relationships among agents; an ability to model trust among agents; an ability to specify requests via hard and soft constraints; and support for privacy-preserving information sharing among agents. InterPol's novel features are demonstrated using examples from health care.

2. INTERPOL FRAMEWORK

The InterPol architecture consists of *agents*, representing *principals*. The agents are heterogeneous and differ in their policies and needs. For simplicity, we assume they share a communication language.

2.1 Agent Interactions

InterPol employs a multiagent referral architecture wherein agent interactions are based on the following mechanism. An agent seeking a service requests some agents from among its *neighbors*. A requested agent may ignore a request, perform the specified service, or give referrals to other agents. An *answer* is a response based on performing the requested service; a *referral* is a response consisting of names of other agents who might provide the requested service.

Figure 1 shows a simple scenario (ignore policies for now), where Alice queries Bob and Charlie for a service. Bob returns an answer, while Charlie refers Gabriel. Alice then queries Gabriel.

2.2 Policy Framework

InterPol goes beyond traditional referral approaches by providing a sophisticated means for specifying interaction policies among the participants. The following examples

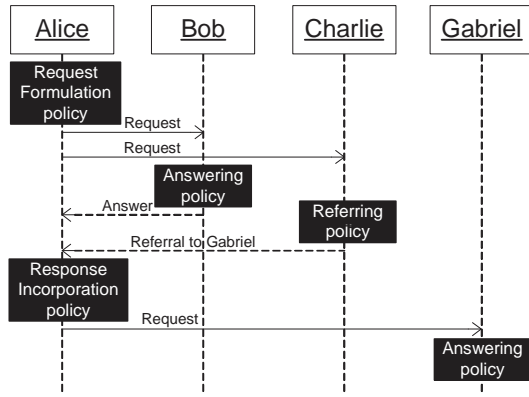


Figure 1: Example referrals scenario

provide a flavor of the kinds of policies supported. A user may specify that his personal information can be shared only with a physician P who has credentials from a local hospital to which the user has revealed personal information and if P is given a referral by the user’s current primary care physician. A user may not want to reveal any private information to any one but his friend. InterPol supports the following kinds of policies.

Request Formulation (RF) Policies An agent applies its *request formulation* policies to decide on what to request and whom to ask.

Response Incorporation (RI) Policies An agent applies its *response incorporation* policies to evaluate the responses and decide on further action.

Answering and Referring Policies An agent, when requested, applies its *answering* and *referring* policies to decide whether and how to provide an answer or a referral.

Figure 1 illustrates these policies for an example scenario.

Enactment. We have implemented *InterPol* to demonstrate the effect that the above approach has on modeling and reasoning about the interactions among agents in a service network. Each agent is implemented around a reasoner (built using the tuProlog interpreter [2]) that handles policies. Each agent has two knowledge bases (KBs): *domain KB* storing knowledge related to the agent’s domains of interest and expertise, and *social KB* storing knowledge about neighbors, agent models, and social relationships. There is a policy base for each kind of policy introduced above: request, answer, referral, and response incorporation. These are executed via the agent’s interpreter. Our agents follow the architecture typical in referral systems, e.g., [8].

2.3 Representation

InterPol uses *Constraint Datalog* [5] to express policies and facts. By convention, variable names begin with an uppercase letter and constant names with a lowercase letter.

Facts and Policies. An agent’s social knowledge captures social relationships and interactions independent of any domain. An agent’s domain knowledge captures an agent’s expertise in its domains of interest. In InterPol, the social and domain knowledge of an agent comprises sets of facts and rules. These knowledge bases (KBs) are dynamic: facts and rules may be continually added or retracted. Social facts include neighborhood relations, agent models, referral

knowledge, and so on. Policies are logic rules.

For example, Listing 1 shows facts and policies in Alice’s KB. These indicate that: Dave is a physician specializing in cardiology, Alice likes Charlie for the *findPhysician* service, and a fact (illustrating the use of a nested predicate) that the first fact is public. Alice’s referring policy allows her to refer any agent Y for a service P if she likes that agent.

Listing 1: Facts and policies in Alice’s KB

```
physician(dave, cardio).
likes(alice, charlie, findPhysician).
factType(physician(dave, cardio), public).
refer(P, alice, Y) :- likes(alice, Y, P).
```

Queries and Tacks. Let’s first consider a simple form of a request, which consists of a *query rule* whose head is the predicate *ask* applied to some variables. The variables free in the head are used along with other variables in the body of the rule. Listing 2 shows a simple request.

Listing 2: Alice’s simple request

```
[ask(X) :- physician(X, cardio),
  medicalSchool(X, duke),
  certifiedBy(X, abms),
  experience(X, Y), Y > 10]
```

To improve the effectiveness and efficiency of interactions, InterPol supports requests that consist of a query rule and a list of *tacks*. Each tack is a conjunction of one or more clauses. A tack expresses a preference of the requester. When an agent responding to a request is able to accommodate a specified tack, it facilitates the requester pruning the search space and reducing the communication overhead.

Listing 3: Alice’s request with tacks

```
[ ask(X) :- physician(X, cardio),
  { medicalSchool(X, duke),
    certifiedBy(X, abms),
    experience(X, Y), Y > 10 } ]
```

Listing 3 shows a request sent out by Alice. Alice is looking for a physician specializing in cardiology. She has soft preferences expressed in three tacks, respectively, that the physician should have studied at Duke Medical School, be certified by the ABMS (American Board of Medical Specialties), and have more than 10 years’ experience.

Answers and Referrals. A response returned by an agent is either an answer or a referral. For a simple request, each solution is a vector of bindings of the variables in the *ask* of the given query to constants that satisfy the query rule. A *referral* is a set of facts describing the agents referred. For a request with tacks, each solution has two parts: (1) a vector of bindings of the variables in the *ask* of the given query to constants that satisfy the query rule and (2) a list of *remarks* in the same order as the tacks in the given request. Each remark on a variable binding states whether the corresponding tack is true (T) or not (F) for that binding.

3. APPLYING INTERPOL

To best capture interactions in multiagent service networks, InterPol incorporates a conceptual model and predicates for interactions, social relationships, trust evaluations, and privacy and utility management. The following are the major scenarios relevant in multiagent literature that motivate the development of a rich vocabulary for the policy specification language in InterPol.

Accommodating Privacy, Trust, and Social Relationships. Policy-based approaches are natural for privacy. InterPol provides two low-level primitives for handling privacy. First, it allows a fact or a rule in the KB to be marked with its *visibility* (*public* or *private*). Second, InterPol supports a notion of *privacy level* both with respect to services and agents. The concepts of visibility and privacy level enable formulating precise answering policies that restrict revealing private information to certain agents. InterPol models these concepts using the predicates *factType* and *ruleType* to indicate the visibility, and using the predicates *servicePrivacyLevel* and *agentPrivacyLevel* (values in the range $[0, 1]$) to specify the privacy levels of service and an agent, respectively. Here a privacy level of 0 (1) means highly private (public).

We model *trust* in relational terms: a *trustor* trusts a *trustee* with respect to a particular service. For example, we may trust a cardiologist for all heart-related problems but not for other ailments. *Social trust* is based on the relationships among the agents. We describe an example of a generic means to evaluate relationships, which provides the heart of evidence-based reasoning. Social network analysis models trust in the presence of social relationships based on evaluating the participants' experiences [7]. The knowledge of these relationships at various strength levels can feature in an agent's policies to evaluate trust among agents. InterPol captures the strength I of a relationship R via a measure $rStrength(R, I)$, which takes values in the interval $[0, 1]$.

Strategies for Requests. An agent's requests can potentially reveal too much information, e.g., about the agent's true needs. A public request modifies a true, private request so as to hide some of the private information. To formulate privacy preserving queries, an agent must infer public requests from its private needs. There are two main ways of accomplishing this. In *generalization*, a weaker request is revealed. For example, a request for a physician for skin allergy can be replaced by a request for a physician who treats any allergy. In the *association* approach, a request that is a sibling of the actual (private) need is used. For example, the agent can request a dermatologist, based on the association between skin allergy and dermatology.

4. DISCUSSION

Our work is motivated by the needs of emerging P2P information systems. An important and natural class of such systems arise in health care information management. Our examples are inspired by those studied in the EU project Artemis [3], which is developing an approach to enable the sharing of health care information across organizational and sometimes national boundaries. Social relationships apply naturally here. A patient would stop seeing a physician with whom his interactions were not effective. And he would form additional relationships based on his evolving needs. Privacy is an important concern in health care and policies are natural for privacy management. InterPol was evaluated on a health care scenario where agents request each other for names of physicians meeting various criteria. Here, an answer typically involves names of physicians, sometimes with additional information about them. And, a referral typically is to an agent who might be able to provide the names of some physicians meeting the specified criteria.

InterPol shows how its algorithms can be realized over a conventional Prolog engine. It provides a rich vocabulary to

enable to proper expression of policies, and supports various heuristics by which agents can interact with each other. Future work will consider enhancing the algorithms for evaluating policies to support better exchange of information among the agents to perform cooperative search.

Related Work. Policies are widely used for access control and trust management in distributed systems. Traditional policy languages provide primitives for published credentials or certificates, which attest to the identity or other attributes of the given party. Of the several policy specification languages, two are particularly important. *Rei* is a policy language implemented in Prolog for pervasive environments and uses Web Ontology Language (OWL) to represent entities, access types, policy objects, speech acts, policies, and metapolicies [4]. *PeerTrust* has an expressive policy and trust negotiation language based on first order Horn rules which form the basis for logic programs [6]. *Rei* does not model the privacy preserving policies like in InterPol and PeerTrust. Like in PeerTrust, trust between entities in InterPol is built over time, but unlike the dynamic exchange of certificates in PeerTrust, trust in InterPol depends on the quality of the answers or referrals provided by the entities, and the trust models generated by the policy framework.

5. REFERENCES

- [1] R. Bonnell, M. Huhns, L. Stephens, and U. Mukhopadhyay. MINDS: Multiple intelligent node document servers. In *Proceedings of the 1st IEEE International Conference on Office Automation*, pages 125–136, 1984.
- [2] E. Denti, A. Omicini, and A. Ricci. tuProlog: A light-weight Prolog for Internet applications and infrastructures. In *Proceedings of 3rd International Symposium on Practical Aspects of Declarative Languages*, volume 1990 of *LNCS*, pages 184–198. Springer-Verlag, Jan. 2001.
- [3] A. Dogac, G. Laleci, S. Kirbas, Y. Kabak, S. Sinir, and A. Yildiz. Deploying semantically enriched web services in the healthcare domain. *Information Systems Journal (Elsevier Science)*, 2005.
- [4] L. Kagal, T. Finin, and A. Joshi. A policy language for a pervasive computing environment. In *Proceedings of 4th International IEEE Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 63–74, June 2003.
- [5] N. Li and J. C. Mitchell. Datalog with constraints: A foundation for trust management languages. In *Proceedings of 5th International Symposium on Practical Aspects of Declarative Languages*, Jan. 2003.
- [6] W. Nejdl, D. Olmedilla, and M. Winslett. PeerTrust: Automated trust negotiation for peers on the semantic web. In *VLDB Workshop on Secure Data Management (SDM)*, volume 3178 of *LNCS*, pages 118–132. Springer-Verlag, Aug. 2004.
- [7] J. Sabater and C. Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 475–482, 2002.
- [8] M. P. Singh, B. Yu, and M. Venkatraman. Community-based service location. *Communications of the ACM*, 44(4):49–54, Apr. 2001.